ROBOFUN.RO

LECȚIA VIII

CURS GRATUIT

ARDUINO ȘI ROBOTICĂ

Arduino - Comunicare prin Internet

Proiect :
Ultimul tweet al unui utilizator afişat pe un LCD

Ethernet Shield

Prin intermediul acestui shield, Arduino se poate conecta la Internet exact ca un PC obisnuit. Poate functiona in regim de client (accesand alte site-uri web din Internet, asa cum faci tu cand navighezi pe Internet) sau poate functiona in regim de server web (si atunci tu – sau oricine altcineva – il poate accesa de oriunde din Internet folosind un browser obisnuit).

Aplicatiile sunt multe si spectaculoase. Spre exemplu, poti face ca Arduino sa citeasca date de la senzori de mediu (temperatura, presiune atmosferica, umiditate, nivel de monoxid de carbon) si sa le trimita la fiecare 5 secunde prin Internet catre un formular tip Excel creat in Google Docs. La fel de simplu este ca in loc de Google Docs sa folosesti COSM.COM, un serviciu dedicat pentru culegere si stocare de date de la senzori. Sau poti folosi un senzor de umiditate montat intr-un ghiveci, si atunci cand planta nu are suficienta apa, Arduino iti va trimite mesaje pe Twitter. Sau, daca ai un spatiu comun in care se patrunde pe baza de card-uri RFID, atunci Arduino poate anunta pe Twitter prezenta unei anumite persoane (solutie pe care chiar o folosim in acest moment pentru accesul la hacker-space-ul inventeaza.ro din Bucuresti). Sau poti face ca Arduino sa se conecteze la serverul weather.com, sa obtina vremea probabila, si sa o afiseze pe un LCD montat pe oglinda din baie.

Client Web, IP prin DHCP

Exemplul de mai jos demonstreaza o conexiune facuta cu Arduino la serverul google.com, conexiune pe care Arduino o foloseste ca sa caute pe Google termenul "arduino". Rezultatul cautarii (la nivel de HTML) este afisat in Serial Monitor. Inainte de a trece la cod, sa povestim un pic de comunicarea prin Internet. Un browser (cel pe care il folosesti tu ca sa citesti stirile online) face o cerere catre un server (calculatorul aflat la sediul firmei de hosting care stocheaza informatia), iar in urma acestei cereri, serverul raspunde cu un text in format HTML. HTML este un mod de reprezentare a informatiei vizuale, care contine atat textul pe care il citesti tu, cat si elementele de formatare in pagina. Poti vedea exact codul HTML al unei pagini daca folosesti optiunea "view source" din browser. Pe langa HTML, raspunsul serverului mai contine si un text care contine informatii despre raspuns (numit "header HTTP"). Daca atunci cand accesezi o pagina in browser, toata informatia suplimentara legata de formatarea in pagina si de header-e HTTP este deja interpretata si folosita de browser (astfel incat tu vezi doar rezultatul final, informatia vizuala), in cazul in care facei o cerere web cu Arduino nu se intampla asta. La Arduino ajunge

intreaga informatie generata de browser, neprocesata in vreun fel, inclusiv header-ele HTTP. Astfel, ceea ce vei vedea in Serial Monitor in cazul exemplului de mai jos nu va fi chiar ceea ce vezi in browser cand faci o cautare pe Google, dar informatia va fi exact aceeasi.

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = {  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress server(209,85,148,101); // Google

EthernetClient client;

void setup() {
  Serial.begin(9600);

  if (Ethernet.begin(mac) == 0) {
    Serial.println("Nu s-a reusit initializarea placii de retea folosind
DHCP");
  }

  delay(1000);
  Serial.println("conectare in progress...");

  if (client.connect(server, 80)) {
    Serial.println("conectat");
    client.println("GET /search?q=arduino HTTP/1.0");
    client.println();
  }
  else {
    Serial.println("conectare esuata");
  }
}

void loop() {
  if (client.available()) {
    char c = client.read();
    Serial.print(c);
  }

  if (!client.connected()) {
    Serial.println();
    Serial.println("deconectare acum.");
    client.stop();

    for(;;)
      ;
  }
}
```

Primul lucru de remarcat in codul sursa de mai sus este declaratia adresei MAC. Orice dispozitiv conectat intr-o retea este identificat in mod unic de adresa sa MAC. Aceasta este un numar unic alocat de producator, care permite adresarea dispozitivului respectiv. Daca te uiti pe shield-ul tau Ethernet vei

vedea acest numar de identificare scris pe un sticker. Este bine sa folosesti acest numar de identificare ori de cate ori scrii cod pentru shield-ul Ethernet respectiv. Cu toate ca pana acum am spus ca acest numar este "unic", lucrurile nu sunt chiar asa de stricte. Este absolut necesar sa fie unic dar nu neaparat la nivelul intregii retele Internet, ci doar in reteaua locala (pana la primul router) in care este cuplat dispozitivul. Exista sanse sa functioneze corect (aproape) orice valori ai seta in codul de mai sus pentru adresa MAC (spun "aproape" pentru ca exista o serie de reguli care determina o adresa MAC valida, dar care nu sunt neaparat respectate). Ca si concluzie, cel mai bine este sa declari ca adresa MAC exact ceea ce scrie pe shield. Daca nu scrie, sau daca sticker-ul s-a rupt, poti pune si alte valori si exista sanse mari sa functioneze corect.

Al doilea lucru este adresa IP. Pe langa adresa MAC de care am discutat mai sus, fiecare dispozitiv conectat in Internet mai are asociata si o adresa IP. Daca este prima data cand te intalnesti cu aceste concepte, probabil ca ti se pare totul anapoda si te intrebi de ce atunci cand a fost gandit Internetul n-au dat fiecarui dispozitiv o adresa unica si s-au complicat in halul asta. Exista o serie de motive, dar nu voi insista asupra lor aici. Deocamdata este suficient sa acceptam ca pe langa adresa MAC, pentru fiecare dispozitiv mai exista si o adresa IP. Ca sa poate fi vizibil in Internet (adica oricine sa-l poata accesa din Internet), atunci adresa IP a dispozitivului trebuie sa fie unica (si aici chiar trebuie, nu mai merg lucrurile ca la adresa MAC). In cazul nostru, in care un Arduino care se conecteaza la serverul Google, putem identifica doua adrese IP – o adresa IP pentru serverul Google (anume " 173,194,33,104") si o adresa IP pentru Arduino. Adresa IP a serverului Google este unica in intreg Internetul, pentru a fi accesibil de oriunde. In cazul adresei IP pentru Arduino, este suficient ca adresa sa fie unica doar la nivelul retelei locale (pana la primul router – ceea ce de obicei inseamna casa sau biroul tau), pentru ca in aceasta situatia noi folosim Arduino drep client doar (adica Arduino nu este accesat de catre cineva din Internet, ci el acceseaza). Din acest motiv, putem lasa alocarea unei adrese IP pentru placa Arduino pe seama router-ului, fara sa ne mai batem noi capul cu ea (procedeul se numeste alocare prin DHCP). Evident, acest lucru presupune ca avem un router capabil sa ofere adrese prin DHCP in reteaua locala ! Daca nu ai, atunci va trebui sa setezi tu o adresa IP statica pentru placa de retea (si in exemplu urmator vom face asta).

Inca o remarca utila este ca adresa IP a serverului Google se poate schimba in timp. In momentul in care eu scriu aceste randuri, exemplul de mai sus functioneaza perfect. Se poate intampla insa ca pana cand tu le vei citi, inginerii de la Google sa fi modificat deja adresa IP a serverului, iar tie sa nu-ti functioneze corect exemplul. Un browser web obisnuit, ca sa obtina intotdeauna adresele IP corecte utilizeaza un serviciu numit "DNS" (serviciu pe care il poate folosi si Arduino, dar intr-un exemplu urmator). Deocamdata, daca nu-ti functioneaza exemplul de mai sus, va trebui sa determini si sa actualizezi manual adresa serverului Google. Pentru aceasta, deschide o consola command prompt pe PC (daca esti pe Windows, apasa "Start", "Run", scris

"cmd" si apasa Enter). In consola deschisa scrie "ping google.com". Vei vedea ca raspuns IP-ul serverului Google, ca mai jos.

In sfarsit, daca totul merge bine, atunci ar trebui sa vezi in Serial Monitor o serie de caractere care defileaza. Acestea sunt raspunsul serverului Google la cautarea ta (acel HTML pe care browserul il interpreteaza si il afiseaza intr-un format vizual adecvat).

Exemplul pe care tocmai l-am vazut nu are o utilitate clara in sine exact in forma aceasta, dar este baza pentru orice proiect care presupune extragere de informatia din Internet.

Client Web, IP static

Sa presupunem ca nu ai in casa un router capabil sa aloce adresa IP prin DHCP. In aceasta situatie, vei vedea in Serial Monitor un mesaj de eroare in momentul in care vei incerca sa rulezi exemplul de mai sus. In aceasta situatie, va trebui sa declari tu manual o adresa IP statica pentru shield-ul Ethernet, ca mai jos.

```
#include <SPI.h>
#include <Ethernet.h>

byte mac[] = {  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress server(209,85,148,101); // Google
IPAddress arduinoIP(192,168,0,99); // Arduino

EthernetClient client;

void setup() {
  Serial.begin(9600);

  if (Ethernet.begin(mac, arduinoIP) == 0) {
    Serial.println("Nu s-a reusit initializarea placii de retea folosind o
adresa IP statica");
  }

  delay(1000);
  Serial.println("conectare in progress...");

  if (client.connect(server, 80)) {
    Serial.println("conectat");
    client.println("GET /search?q=arduino HTTP/1.0");
    client.println();
  }
  else {
```

```
    Serial.println("conectare esuata");
  }
}

void loop(){
      //EXACT LA FEL CA IN EXEMPLUL PRECEDENT
}
```

Singurele linii diferite sunt cele doua linii marcate cu **bold**. Prima linie defineste o adresa IP pentru shield-ul Ethernet, iar cea de-a doua linie o utilizeaza in sectiunea de initializare a shield-ului. Ca sa alegi o adresa IP valida pentru reteaua ta, va trebui sa determini mai intai ce adresa IP are calculatorul tau, ca sa poti da si pentru Arduino o adresa IP din aceeasi clasa. Acest lucru se face (in Windows) apasand "Start", apoi "Run", apoi tastand "cmd" si Enter. In consola deschisa tasteaza "ipconfig". Vei observa (daca nu ai foarte mult ghinion) o linie care zice "IP Address : 192.168.0.2" sau ceva similar. Aceasta este adresa IP a calculatorului tau. In cele mai multe cazuri, tu va trebui sa modifici ultima cifra astfel incat adresa nou obtinuta sa nu mai fie utilizata de nimeni in reteaua ta locala. Depinzand de la caz la caz, acest lucru ar putea fi simplu sau mai complicat. Incearca sa alegi un numar mai mare (99, 149, 253 sunt exemple bune). Daca totusi nu reusesti sa gasesti o adresa IP valida, atunci o varianta ar fi sa apelezi la ajutorul unui amic care se pricepe la retele de calculatoare si sa ii ceri sa-ti indice o adresa IP libera in reteaua ta. In final, vei obtine acelasi rezultat ca in exemplul de mai sus, doar ca acum adresa IP pentru placa Arduino este setata de tine.

Server Web

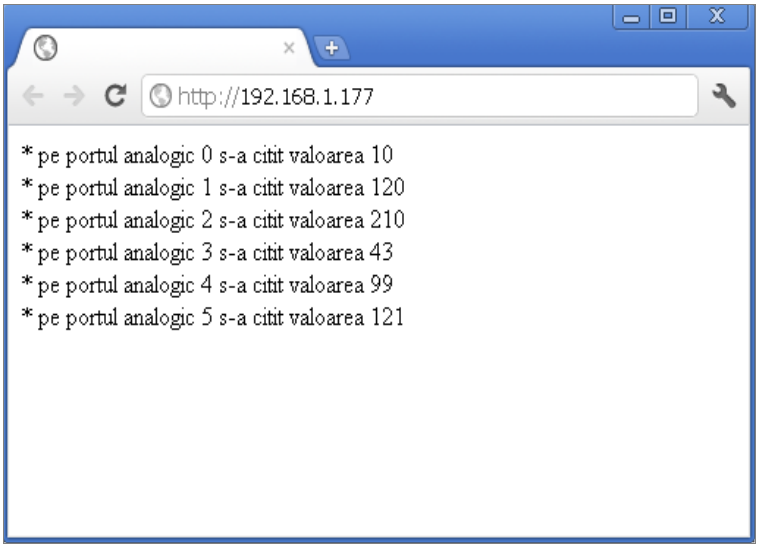
Acum ca ai reusit sa faci placa Arduino sa citeasca date din Internet, urmatorul pas este sa faci un server care sa poata fi accesat de alti utilizatori din Internet, interesati sa vada informatie culeasa de placa ta Arduino. Inainte de a incepe, este bine sa stii ca in cele de urmeaza voi prezenta in detaliu tot ce trebuie sa faci pentru a avea un server functional pe placa Arduino (si care sa functioneze cel putin accesat dintr-un browser de pe laptopul tau). Ca serverul sa fie accesibil chiar din Internet, pentru oricine, lucrurile sunt putin mai complicate la nivel de configurari de router si retea locala (nu la nivelul Arduino). Cu aceste configurari de router va trebui sa te descurci singur. Din fericire, exista foarte multe tutoriale in acest sens pe Internet (poti incerca o cautare pe Google cu "access home server from internet" sau "cum accesez serverul de acasa").

```
#include <SPI.h>
#include <Ethernet.h>
```

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress ip(192,168,1, 177);

EthernetServer server(80);

void setup() {
  Serial.begin(9600);

  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("adresa IP a server-ului este: ");
  Serial.println(Ethernet.localIP());
}


void loop() {

  EthernetClient client = server.available();
  if (client) {
    Serial.println("conectare client nou");
    boolean currentLineIsBlank = true;
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        Serial.write(c);
        if (c == '\n' && currentLineIsBlank) {
          client.println("HTTP/1.1 200 OK");
          client.println("Content-Type: text/html");
          client.println("Connnection: close");
          client.println();
          client.println("<!DOCTYPE HTML>");
          client.println("<html>");
          client.println("<meta http-equiv=\"refresh\" content=\"5\">");
          for (int analogChannel = 0; analogChannel < 6; analogChannel++) {
            int sensorReading = analogRead(analogChannel);
            client.print("* pe portul analogic ");
            client.print(analogChannel);
            client.print(" s-a citit valoare ");
            client.print(sensorReading);
            client.println("<br />");
          }
          client.println("</html>");
          break;
        }
        if (c == '\n') {
          currentLineIsBlank = true;
        }
        else if (c != '\r') {
          currentLineIsBlank = false;
        }
      }
    }
    delay(1);
    client.stop();
    Serial.println("clientul a incheiat sesiunea");
  }
}
```

Deja am explicat in exemplele precedente ce inseamna adresa MAC si adresa IP, asa ca nu vom mai relua aici. Prima linie interesanta (si prima notiune nou introdusa), este cea care declara un server web pe portul 80. Un server web functioneaza pe unul sau mai multe porturi. Un port este un identificator pentru un anumit serviciu pe un anumit calculator fizic. Astfel, pe un calculator fizic putem avea un server web pe portul 80, si un server de baze de date care raspunde pe portul 3306. Portul 80 este ales de obicei pentru serverele web, si este presupus in mod obisnuit in browsere cand vrem sa accesam un site.

Mai departe, in rutina loop se asteapta conectarea unui client web (adica un vizitator oarecare din Internet) (apelul "server.available()" astepta, fara sa treaca mai departe, pana cand se conecteaza un vizitator la server). Cand acesta s-a conectat, este generata o instanta "client", care va face toata treaba pentru acel vizitator. Mai exact, va incepe prin a genera header-ele HTTP (necesare pentru ca browser-ul vizitatorului sa stie ce sa faca cu informatia pe care o primeste de la server – in cazul nostru, sa o afiseze ca text). Urmeaza apoi un ciclu "for" care citeste toate cele 6 porturi analogice ale placii Arduino si le trimite catre browser-ul vizitatorului.



Daca deschizi un browser si accesezi serverul pe care tocmai l-ai creat, vei vedea un rezultat similar cu cel din imagine.

Dat fiind faptul ca nu ai nimic conectat la porturile analogice ale placii Arduino, valorile citite sunt aleatoare. Daca insa vei conecta un senzor de lumina brick sau un senzor de temperatura brick, vei vedea imediat ca valorile capata sens.

Asa cum spuneam si la inceputul acestei sectiuni, ca sa accesezi placa

Arduino chiar din Internet (si nu de pe laptop cum am testat mai sus), mai trebuie doar sa setezi router-ul local ca atunci cand cineva il acceseaza din afara pe un port pe care ti-l alegi tu, sa trimita cererea catre IP-ul alocat placii Arduino pe portul 80. Pentru detalii despre cum anume sa faci acest lucru, poti incerca o cautare pe Google cu "access home server from internet" sau "cum accesez serverul de acasa".

Client Web, cu DNS

In ambele exemple de mai sus in care am folosit placa Arduino ca sa citim informatie din Internet am avut adresa IP a server-ului scrisa in clar in codul sursa. Dupa cum am vazut, acest lucru poate cauza probleme atunci cand adresa IP a server-ului s-a schimbat intre timp. Exista un singur lucru care nu se schimba, si acesta este domeniul server-ului (numele acestuia – "google.com"). Trecerea de la domeniu la adresa IP se cheama "rezolvarea domeniului" si este asigurata de o serie de servere dedicate prezente in Internet, numite servere DNS. Arduino este capabil sa le utilizeze ca sa obtina IP-ul unui anumit server pe baza domeniului acestuia, ca mai jos. Este exact acelasi exemplu ca mai sus (cautam pe Google termenul "arduino"), dar de data asta folosind serviciul DNS.

```
#include <SPI.h>
#include <Ethernet.h>
byte mac[] = {  0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02 };
char serverName[] = "www.google.com";

EthernetClient client;

void setup() {
  Serial.begin(9600);
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Eroare in configurarea folosind DHCP.");
  }
  delay(1000);
  Serial.println("conectare in progres...");


  if (client.connect(serverName, 80)) {
    Serial.println("conectat");
    client.println("GET /search?q=arduino HTTP/1.0");
    client.println();
  }
  else {
    Serial.println("conexiune esuata");
  }
}

void loop() {
//LA FEL CA IN EXEMPLELE PRECEDENTE
```

```
}
```

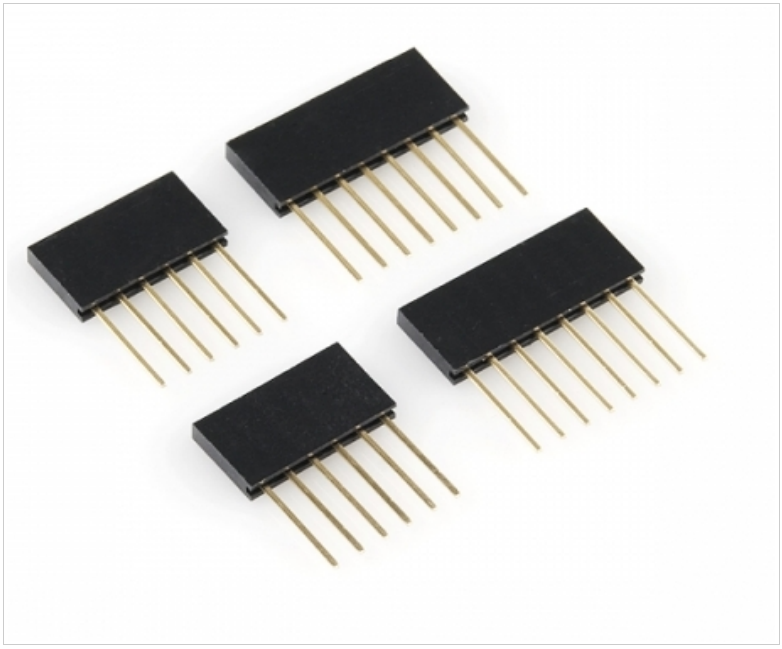
Multumita librariei excelent scrise, modificarile sunt minimale (doar cele doua linii marcate cu **bold**), toata implementarea fiind facuta direct in librarie.

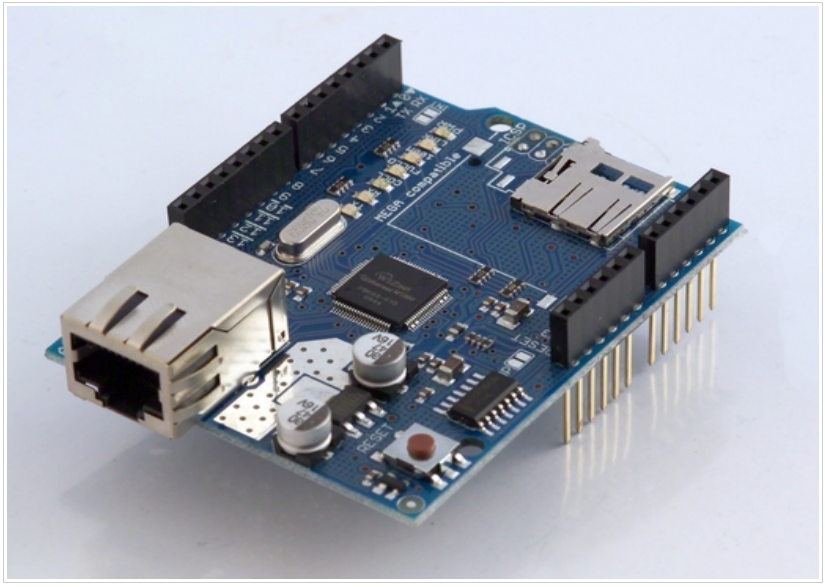
Ultimul tweet al unui user Twitter afisat pe LCD Shield, folosind Ethernet Shield (sau Arduino Ethernet)

Acest exemplu demonstreaza utilizarea unui Ethernet Shield (sau al unui Arduino Ethernet) pentru a afisa ultimul tweet al unui utilizator Twitter pe LCD. Poti folosi orice fel de LCD doresti. In acest exemplu vom folosi un LCD shield, pentru simplitatea conectarii. Ai la dispozitie doar 32 de caractere pe acest LCD, asa ca vom folosi facilitatea de scroll, pentru a vedea intreg tweet-ul. LCD Shield-ul foloseste pinii 2, 3, 4, 5, 6 si 7, in timp ce Ethernet Shield-ul foloseste pinii 10, 11, 12 si 13, asa ca din fericire nu avem nici un conflict de pini.

Din cauza faptului ca Ethernet Shield este mai inalt decat de obicei,

datorita mufei Ethernet, daca incercam sa infigem direct LCD Shield-ul, s-ar putea ca mufa Ethernet Shield-ului sa atinga anumite contacte de pe spatele LCD Shield-ului. Pentru a evita acest lucru, vom folosi un set de pini ca cei din imaginea de mai sus. Vom infige mai intai pinii in Ethernet Shield si apoi in pini vom infige LCD Shield-ul. In acest fel, ne vom asigura ca exista suficient spatiu intre Ethernet Shield si LCD Shield.

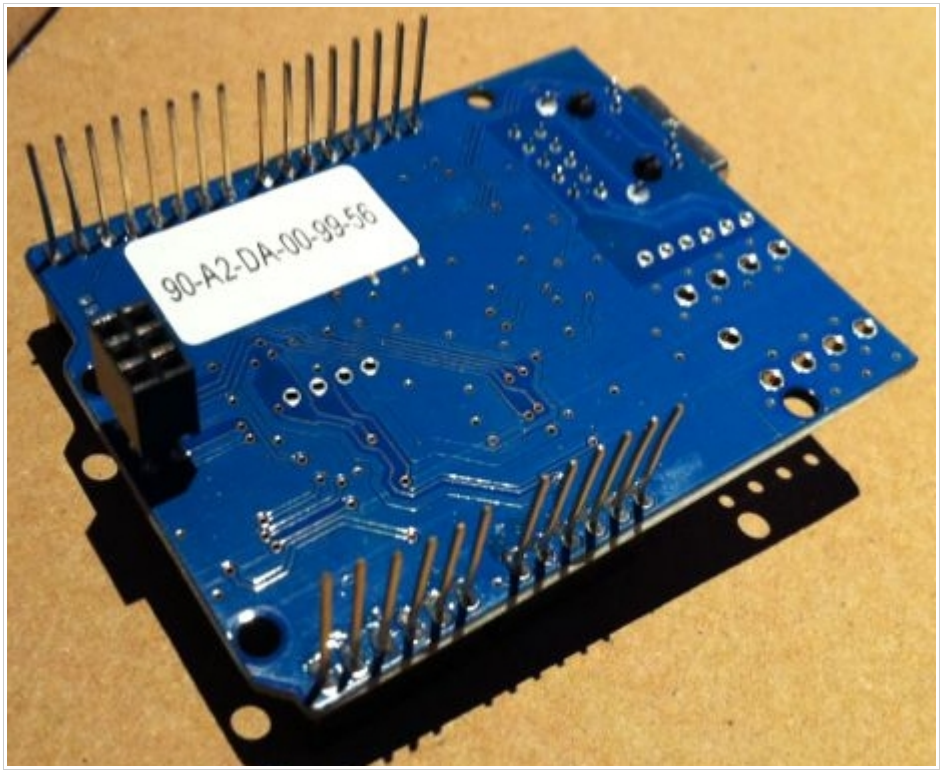




```
#include <SPI.h>
#include <Ethernet.h>
#include <LiquidCrystal.h>
#include <Wire.h>
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);


char username[] = "arduino";

byte mac[] = {
  0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x01 };
IPAddress ip(192,168,1,20);

EthernetClient client;

const unsigned long requestInterval = 60000;
char serverName[] = "api.twitter.com";

boolean requested;
unsigned long lastAttemptTime = 0;

String currentLine = "";
String tweet = "";
boolean readingTweet = false;

void setup() {

  currentLine.reserve(256);
  tweet.reserve(150);

  Serial.begin(9600);

  Serial.println("Incerc sa obtin o adresa IP folosind DHCP:");
  if (!Ethernet.begin(mac)) {
    Serial.println("esec in obtinerea unei adresa IP prin DHCP, vom folosi
adresa IP setata manual");
    Ethernet.begin(mac, ip);
  }

  Serial.print("Adresa IP:");
  Serial.println(Ethernet.localIP());

  connectToServer();

  lcd.autoscroll();
  lcd.clear();
}



void loop() {
  if (client.connected()) {
    if (client.available()) {
      char inChar = client.read();

      currentLine += inChar;
```

```
        if (inChar == '\n') {
          currentLine = "";
        }
        if ( currentLine.endsWith("<text>")) {
          readingTweet = true;
          tweet = "";
        }
        if (readingTweet) {
          if (inChar != '<') {
            tweet += inChar;
          }
          else {
            readingTweet = false;
            Serial.println(tweet);
            lcd.clear();
            lcd.print(tweet);
            lcd.autoscroll();
            client.stop();
          }
        }
      }
    }
    else if (millis() - lastAttemptTime > requestInterval) {
      connectToServer();
    }
  }

  void connectToServer() {
    Serial.println("conectare la server...");
    if (client.connect(serverName, 80)) {
      Serial.println("lansez request HTTP...");
      client.print("GET /1/statuses/user_timeline.xml?screen_name=");
      client.print(username);
      client.println("&count=1 HTTP/1.1");
      client.println("HOST: api.twitter.com");
      client.println();
    }
    lastAttemptTime = millis();
  }
```

Constanta *username* defineste userul Twitter pentru care ne intereseaza sa obtinem ultimul tweet. Constanta MAC este un identificator unic al placii tale Ethernet. Este foarte probabil sa il gasesti scris chiar pe placa (ca in poza de mai jos). Chiar daca shield-ul tau nu are MAC-ul scris pe el, nu este nici o problema. Poti sa-l folosesti pe cel din codul de mai sus fara probleme. Singura situatie in care s-ar putea intampla sa ai probleme este aceea in care un alt echipament din reteaua ta (de exemplu un laptop) sa aiba exact acelasi MAC (lucru care nu este foarte probabil).



Constanta *ip* este adresa IP a placii, pentru cazul in care nu s-a reusit obtinerea unei adresa IP in mod automat, prin DHCP. Pentru mai multe detalii despre adrese IP si DHCP, vezi sectiunea dedicata Shield-ului Ethernet.

requestInterval defineste intervalul intre doua actualizari succesive, in milisecunde. Esti liber sa folosesti ce interval doresti, dar nu mai putin de 20000, pentru a oferi un interval de 20 de secunde minim pentru a se finaliza executia unei actualizari.

api.twitter.com este URL-ul server-ului Twitter catre care vom lansa cererile HTTP.

In functia *setup* se incearca obtinerea unei adrese IP in mod automat, folosind protocolul DHCP. Daca nu s-a reusit acest lucru, se foloseste adresa IP setata manual la inceputul programului. c*onnectToServer* lanseaza cererea HTTP catre server-ul Twitter (include in cererea HTTP si username-ul setat la inceputul programului).

Imediat ce s-a terminat *connectToServer*, in functia *loop* incepe citirea caracterelor trimise de server. Se citeste caracter cu caracter, iar in variabila *currentLine* se concateneaza caracterele pentru a obtine intreaga linie (sfarsitul

unei linii de text este marcat prin caracterul "\n"). Atunci cand linia curenta este "<text>", inseamna ca incepand din acest punct, pana cand intalnim textul "</text>" avem continutul tweet-ului care ne intereseaza. Imediat ce am intalnit caracterul "<" (care marcheaza inceputul pentru </text>) tweet-ul s-a incheiat. Putem afisa tweet-ul pe LCD (*lcd.print(tweet)*) si putem inchide conexiunea la server *(client.stop())*.

Dupa ce s-au scurs *requestInterval* milisecunde, este apelata din nou functia *connectToServer*, si procesul se reia.

Daca Shield-ul LCD 16X2 ti se pare prea mic, poti folosi orice alt LCD iti place. Intr-o lectie viitoare vom prezenta exact acelasi proiect folosind insa un Wifly Shield in loc de Ethernet Shield si un LCD 20X4 pe I2C in loc de LCD-ul 2X16.

Aceasta a fost lectia 8. In final, as vrea sa te rog sa ne oferi feedback asupra acestei lectii, pentru a ne permite sa le facem mai bune pe urmatoarele.

Este vorba despre un sondaj cu 4 intrebari (oricare este optionala), pe care il poti accesa dand click aici.

Sau ne poti contacta direct prin email la contact@robofun.ro .

Iti multumim,

Echipa Robofun.RO